UNITED STATES PATENT APPLICATION

FOR

DISTRIBUTED DELAY PREDICTION OF MULTI-MILLION GATE DEEP SUB-MICRON ASIC DESIGNS

Inventors:

Saket Goyal
Santhanakrisnan Raman
Prabhakaran Krishnamurthy
Prasad Subbarao
Manjunatha Gowda

Sawyer Law Group LLP
2465 E. Bayshore Road
Suite 406
Palo Alto, CA 94303

DISTRIBUTED DELAY PREDICTION OF MULTI-MILLION GATE DEEP SUB-MICRON ASIC DESIGNS

FIELD OF THE INVENTION

The present invention relates to ASIC design methodologies, and more particularly to a method and system for predicting delays in multi-million gate sub-micron ASIC designs.

BACKGROUND OF THE INVENTION

With innovative technologies, ASIC manufacturers are able to deliver system complexities up to 15-million gates on a single chip. Complete systems comprising cores, memories, and random logic are integrated on a single piece of silicon. Designers may design and verify a complete system for sign-off by submitting complex system models to electronic design automation (EDA) and verification and floor-planning tools to design and verify the complete system.

Figure 1 is a block diagram illustrating a conventional ASIC design flow. The design flow includes a front-end design process that creates a logical design for the ASIC, and a back-end design process that creates a physical design for the ASIC. The front-end design process begins with providing a design entry 10 for an electronic circuit that is used to generate a high-level electronic circuit description, which is typically written in a Hardware Description Language (HDL) 12. Although many proprietary HDLs have been developed, Verilog HDL and VHDL are the major standards.

The design includes a list of interconnections that need to be made between the cells of the circuit; but physical properties for the interconnects have yet to be determined. Therefore, the designer needs an estimation of physical properties to help determine timing within circuit. Interconnect data from previous designs are used to generate interconnect statistical data to use as the estimation in step 14. The interconnect statistical data is used to create a wire load model 16, which defines the resistance, capacitance, and the area of all nets in the design. The statistically generated wire load model 16 is used to estimate the wire lengths in the design and define how net delays are computed.

The HDL 12 and the wire load model 16 are then input into a logic synthesis tool 18 to generate a list of logic gates and their interconnections, called a "netlist" 20. It is important to use wire load models 16 when synthesizing a design, otherwise, timing information generated from synthesis will be optimistic in the absence of net delays. The timing information will also be inaccurate when a poor wire load model 16 is used.

Next, system partitioning is performed in step 22 in which the physical design is partitioned to define groupings of cells small enough to be timed accurately with wire load models 16 (local nets). The resulting design typically includes many cells with many interconnect paths. A prelayout simulation is then performed in step 24 with successive refinement to the design entry 10 and to

logic synthesis 18 to determine if the design functions properly.

After prelayout simulation 24 is satisfactory, the back-end design process begins with floor planning in step 26 in which the blocks of the netlist 20 are arranged on the chip. The locations of the cells in the blocks are then determined during a placement process in step 28. A routing process makes connections between cells and blocks in step 30. Thereafter, circuit extraction determines the resistance and capacitance of the interconnects in step 32.

After circuit extraction, a parasitic extraction process is performed in which a delay prediction application calculates the net delays across the entire design in step 33. The net delays are then input to a post-layout simulation in step 34, with successive refinement to floor planning 26 as necessary.

Figure 2 is a block diagram illustrating a conventional delay prediction process performed after parasitic extraction. Delay prediction is typically performed by a monolithic software application 40 running on a server 42 or mainframe that estimates the delays across all the gates represented in the netlist 20, and outputs the estimated delays as a standard delay format (SDF) output file 44.

Conventional delay prediction process is slow and resource intensive. For example, delay prediction for a 10 million gate ASIC design may take up to

two days to complete. If the subsequent post layout simulation proves inaccurate or the netlist 20 is changed, then the entire physical design process, including the delay prediction, must be repeated. The process is resource intensive because it requires huge physical memory configuration and extremely fast CPUs to run the delay prediction software application 40. For example, to adequately perform a delay prediction calculation for a 10 million gate design, an enterprise class server with 16 GB of physical memory is required, such as a SUN E4500 server. And because the delay prediction is so resource intensive, the use of other software tools on the server are precluded. Therefore, the delay prediction process is not only time-consuming, but is also expensive in terms of hardware requirements.

Previous attempts have been made to speed the delay prediction process. One approach attempted to increase the efficiency of the sequential delay prediction calculations by identifying performance bottlenecks in the delay prediction application 40 and optimizing them to speed up the overall process. This approach only resulted in minimal improvements and failed to have a significant effect on the overall time for the process 40 to run. Another approach attempted to improve runtime by using a multi-threaded delay prediction application, rather than a single-threaded application. Although multithreading improved runtime somewhat, multithreading was not sufficient for reducing delay prediction runtimes on very large designs. In addition, both approaches still required expensive hardware to run the delay prediction application 40.

Accordingly, what is needed is an improved method for performing delay prediction of multi-million gate sub-micron ASIC designs. The delay prediction process should result in significant time improvements over the monolithic application approach and require less expensive hardware resources. The present invention addresses such a need.

SUMMARY OF THE INVENTION

The present invention provides a method and system for delay prediction of multi-million gate sub-micron ASIC designs. The method and system include automatically partitioning a netlist into at least two logic cones, and running respective instances of a delay prediction application on the logic cones on at least two computers in parallel. According to the system and method disclosed herein, the present invention provides a distributed parallel approach for performing delay prediction that significantly improves total run time over the monolithic approach. For a 10 million gate ASIC design for example, the parallel delay predication of the present invention takes approximately 4-6 hours versus 2 days for the monolithic approach. And because delay prediction is performed on small logic blocks in parallel, the delay prediction application can be run on less expensive workstations, rather than requiring an enterprise class server or mainframe.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram illustrating a conventional design flow for fabricating an ASIC.

Figure 2 is a block diagram illustrating a conventional delay prediction process performed during parasitic extraction.

Figure 3 is a flow diagram illustrating the software components of the parallel delay prediction process in accordance with the preferred embodiment of the present invention.

Figure 4 is a flow chart illustrating the process of performing parallel delay prediction in according to the preferred embodiment of the present invention.

Figure 5 is a flow chart of the findcones program a preferred embodiment of the present invention.

DETAILED DESCRIPTION

The present invention relates to a process for predicting timing delays for deep sub-micron ASIC designs. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiments and the generic principles and features described herein will be readily apparent to those skilled in the art. Thus, the present invention is not intended to be limited to the embodiments shown but is to be accorded the widest scope consistent with the principles and features described

herein.

The present invention provides a method and system for performing delay prediction of a multi-million gate sub-micron ASIC design by partitioning the design so that the delay prediction takes significantly less time than the monolithic approach, and doesn't require huge servers or mainframes to perform the estimation. Instead of running the delay prediction application on a server or mainframes, the present invention partitions the design into smaller pieces so that the delay prediction process may be distributed across widely available and affordable workstations and desktop computers. More specifically, the present invention partitions the netlist into multiple timing-independent blocks of logic called cones, and a delay prediction application is run on each cone independently and in parallel on different computers. Because each delay prediction application now has to process less number of nets, run time decreases drastically for each cone. According to the present invention, the parallel approach to performing delay prediction significantly improves total run time over the monolithic approach and requires less costly hardware resources.

Figure 3 is a flow diagram illustrating the software components of the parallel delay prediction process in accordance with the preferred embodiment of the present invention. According to the present invention, the parallel delay prediction process 148 replaces the monolithic delay prediction application 40, but accepts a netlist 150 as user input, and produces a standard delay format

(SDF) output file 158 as output. Consequently, customers submitting the ASIC design for fabrication will see no difference in the input or output to the process. In a preferred environment, the parallel delay prediction process 148 includes a findcones program 152, multiple instances of the delay prediction application 154a and 154b, which are run on different computers, and merge scripts 156.

Figure 4 is a flow chart illustrating the process of performing parallel delay prediction in according to the preferred embodiment of the present invention. The process begins by accepting the netlist 150 of a deep sub-micron ASIC design as input in step 50. In a preferred embodiment, the netlist 150 includes millions of gates. Referring to both Figures 3 and 4, in step 52 the findcones program 152 partitions the netlist 150 into timing-independent logic cones, which include a global clock cone and multiple radius design cones. Timing-independent means that the timing effect of one logic cone does not propagate to, or affect, another cone in the design in any way. The findcones program 152 identifies timing-independent cones of logic by starting with a particular terminating point and examining all gates and paths feeding into it. The findcones program 152 then traces backwards along these paths of logic to determine timing dependencies.

The findcones program 152 begins by identifying all clock networks, reset pins, and logic controlled by a clock throughout the circuit in order to form the global clock cone. The multiple design cones are then formed by examining the

remaining circuits. For a 10 million gate design, the design is preferably partitioned into 40 design cones.

The findcones program 152 partitions the netlist 150 in a manner that not only produces timing-independent cones, but also produces cones that are balanced. Balanced cones means that each cone produced by the partitioning should have relatively the same number of gates so that the delay prediction applications 154b that are run in parallel can process each logic cone in approximately the same amount of time. For example, assume that a 10 million-gate design is partitioned into 40 logic cones, then each of the 40 logic cones should include approximately 250,000 gates.

After the netlist 150 has been partitioned, delays for the global clock cone are calculated in step 54 using a monolithic delay prediction application 154a. The output of the monolithic delay prediction application 154b is a global clock cone delay output file. Although the delay prediction for the global clock cone could be performed using a parallel process, a parallel process is not necessary because even though the global nets control the whole chip, the logic comprising the global nets is relatively small and may easily be processed by the monolithic delay prediction application 154a.

After the delays for the global clock cone are calculated, the global clock cone delay output file and the design cones are used as input to the parallel

delay prediction applications 154b in step 56. In a preferred embodiment, a series of networked computers are used to run multiple instances of the delay predication applications 154b. Preferably, one computer is allocated to run one instance of the delay prediction application 154b, and each delay prediction application 154b performs delay calculations on one design cone. The delay prediction applications 154b should be started on their respective computers at the same time so that delay output files for each cone are produced at approximately the same time.

After the last delay output file has been created, the radius design cone delay output files and the global clock con delay output file are merged by the merge scripts 156 to produce a final SDF output file 158 in step 58. According to the present invention, the parallel delay prediction process 148 results in two benefits. One benefit is increased customer productivity. Due to partitioning and parallel processing, the parallel delay prediction process 148 requires only 4-6 hours to complete for a 10 million gate ASIC design, which is a significant improvement over the turnaround time for the monolithic approach. Another benefit is that the parallel delay prediction process 148 is scalable and predictable. That is, larger designs may be processed in the same amount of time by adding a proportionate number of computers. For example, delay prediction for a 20 million gate ASIC design may be processed in same time as a 10 million gate ASIC design by doubling the number of computers.

Figure 5 is a flow chart illustrating operation of the findcones program 152 in further detail according to a preferred embodiment of the present invention. The findcones program 152 begins by reading the input netlist 150 and any user options in step 200. In a preferred embodiment, the findcones program 152 reads the netlist 150 using a netlist database infrastructure API. In step 202, all data structures in the netlist 150 related to cone traversal are initialized. Several tables are also created which hold pin information based on pin properties, e.g., a table is created which holds all output pins of flip-flops/memories, as described below.

During the initialization, it is determined which data structures are clock pins, and which are focal points for traversal. A cone is defined as starting from a pin of a flip-flop, memory, or primary output/inout. These starting pins are also called focal points of a cone. Focal points are identified as a pin which does not have arcs to output pins of that cell (flip-flop, memory).

Cone terminating points are primary input and output pins of flip-flops and memories. Thus, traversal starts from the focal point of the cone and all pins are found along the path of traversal until the primary input or output pin of a flip-flop or memory is found. During the traversal, if an output pin of a leaf instance is found, the traversal continues for all input/inout pins which have arcs incident on the pin. If during the traversal, input of a combinational cell (a cell which is not a flip-flop or memory) is hit, the net pin is called a prospective pin, and is placed in

a prospective pin list for the cone. Other pins are placed into a pin list for the cone. Thus, a pin can only be in the pin list or in the prospective pin list of a cone. The prospective pin list is kept to just calculate the interconnect delays for those nets. The delay prediction applications uses the list of prospective pins for a cone and does not write an I/O path delay statement in the output SDF file for those instances when any pin of the cells is in the prospective pin list.

After initialization, the global and clock cones are found in step 204. A global cone is a cone containing all of the input pins of flip-flops and memories which are not part of regular focal points. First, all clock cones are traversed starting from all the clock pins found during initialization. All these pins are part of the global cone. While creating this cone, no pin in the global cone is part of the prospective pin list.

After the global cone is found, the normal design cones are found in step 206. Traversal is started from the focal points and is terminated on the terminating points, as described above. A new design cone is not created until the current design cone has a number of pins that is at least ***(total pins) / (2 x number of computers)***. After finding all the design cones, each design cone has three lists: 1) a pin list, 2) a prospective pin list, and 3) a storage instance list.

After finding all of the design cones, the number of design cones generated may be more than the number of computers available. Therefore, the

design cones may be merged in step 208. In this step, if the number of design cones generated is more than the number of computers available, then the smaller design cones are merged into the bigger design cones to generate a number of design cones equal to the number of available computers.

After merging the design cones, there may be some pins that have not been traversed. Therefore, post processing is performed to place those pins into the global clock cone in step 210. The result of post processing is that the global clock cone may have prospective pins. These prospective pins are traversed in the same fashion as the normal design cones except that the focal points are not flip-flop pins.

After post processing, a clock network is created in all design cones to force ramptime on clock pins of the flip-flops/memories in step 212. As each cone contains the list of storage instances that are in the cone, traversal is started from all the clock pins of those instances and the clock network is created. In this case, only clock and non-leaf pins are inserted in the cone. Leaf instance pins are not inserted, as only nets, which are connected to the clock pin, are needed. A leaf instance list is also maintained for the cone which contains the storage instance (flip-flop and memories). This is done to create the global network so statements to force ramptimes on pins can be specified in a delay prediction control file when the delay prediction application is run on a cone. A delay prediction control file which contains all the pins on which force ramptime

has to be specified is generated for each cone. This file is then given to the delay prediction application 154a that runs the global cone.

After the clock network is created, the prospective pin files, force ramptime control files, and cone statistic files are printed for each cone in step 214. A file describing the pin distribution for each cone is also written to help understand how well the partitioning has been performed. The global clock cone object is transformed into a writer cone object and a write netlist method is invoked to write the cone in the form of a hierarchical netlist in step 216. In this step, the global clock cone is first converted into a writer cone. For all the pins (pin list and prospective pin list), a hierarchy is created and all names except the module names, are retained.

All other logic cones are also transformed into writer cones in step 218. Since each cone is independent of the other, they are processed in parallel by using multithreading techniques. The findcones program 152 then generates a list of files for each cone in step 220. In a preferred embodiment, the files include a Verilog netlist, the prospective pin list file, and the force ramp time pin list file.

A method and system for providing distributed delay prediction for multimillion gate deep sub-micron ASIC designs has been disclosed. The present invention has been described in accordance with the embodiments shown, and one of ordinary skill in the art will readily recognize that there could

be variations to the embodiments, and any variations would be within the spirit and scope of the present invention. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.